



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Composer Help

Business Rule Common Block

Business Rule Common Block

Contents

- **1 Business Rule Common Block**
 - 1.1 Business Rules
 - 1.2 Business_Rules_Preferences
 - 1.3 Business Rule Templates
 - 1.4 Genesys Rules System Architecture
 - 1.5 Type of Rules
 - 1.6 Business Rule Block
 - 1.7 Name Property
 - 1.8 Block Notes Property
 - 1.9 Business Rule Package Property
 - 1.10 Facts Property
 - 1.11 Rules Engine URL
 - 1.12 Exceptions Property
 - 1.13 Condition Property
 - 1.14 Logging Details Property
 - 1.15 Log Level Property
 - 1.16 Enable Status Property
 - 1.17 Interaction ID Property
 - 1.18 Output Result Property
 - 1.19 Business Rules Block Runtime Configuration
 - 1.20 Working With Returned Data
 - 1.21 ORS Extensions Property

Business Rules

Composer interfaces with the Genesys Rules Engine, which is part of the [Genesys Rules System](#). A business rule is an external piece of business logic, which can be customized, and then invoked by Genesys applications. Here is an example business rule for a bank: IF product = 'mortgage' and loanAmount >=200000 THEN TTSMsg = 'You must have a credit score of 300 or great to qualify for this loan.' To simplify rule creation, the Genesys Rules System uses [Rule Templates](#). These are initially created by developers and IT professionals. A Composer-compatible plug-in is available for developing business Rule Templates. This plug-in is provided as part of the Genesys Rules System. For information on installing the plugin, refer to the *Genesys Rule System 8.1 Deployment Guide*. See Chapter 2, Installation. Once validated and deployed, Rule Templates are available for customization in the Genesys Rules Authoring Tool GUI. Business analysts then use the templates to create related sets of business rules called Rule Packages. Packaging rules together allows the business analyst to define which rules will support a particular application. You can use Composer's Business Rule block to request the Genesys Rules Engine to execute a [Rule Package](#) in a routing workflow or voice callflow and write the results back to a variable. A [business rule preference](#) specifies the Genesys Rules Authoring Tool server to work with. Find information on using the business rules GUI in the following documents:

- *Genesys Rules System 8.1 Deployment Guide*
- *Genesys Rules System 8.1 Rules Authoring Tool Help*
- *Genesys Rules System 8.1 Rules Development Tool Help*

Note: In the Genesys 8.1 release, the Genesys Rules System is packaged only with the intelligent Workload Distribution product and the Conversation Manager product.

Business_Rules_Preferences

The preferences entered here are used in the Business Rule block, [Business Rule Package property](#). To set Business Rules Preferences:

1. Select **Window > Preferences > Composer > Business Rules**.
2. Configure the connection to the Genesys Rules Authoring Tool (GRAT) server by entering the following fields:
 - **GRAT Server.** Enter the address of the Application server hosting the GRAT Server. When using the Business Rule Package property in the Business Rule block, Composer will connect to this server to query information about packages and rules. Example only: [http://ca-to-lennon:8080](#).
 - **Server Path.** Enter the name of the web application deployed as the GRAT. For example, if you have the GRAT running at [http://ca-to-lennon:8080/genesys-rules-authoring](#), then the GRAT server is [http://ca-to-lennon:8080](#) and the Server Path is /genesys-rules-authoring.
 - **Tenant.** To obtain a list of Rule Packages, Composer will query the GRAT server using an HTTP request to [http://{server-address:port}/tenant/packages](#). Enter the name of the tenant as defined in the Configuration Database.

- **Username.** Enter the username defined in the Configuration Database for logging into the GRAT server.
- **Password.** Enter the password defined in the Configuration Database for logging into the GRAT server.
- **Genesys Rules Engine (Optional). GRE URL.** Enter the URL for the GVP Debugger to use when starting a call. The GRE URL will be passed to the VXML application in the SIP URL. If set, this value will be passed to the voice or routing application and will override the value set in the Rules Engine URL property of the **Business Rule Block** (see that section below).

Business Rule Templates

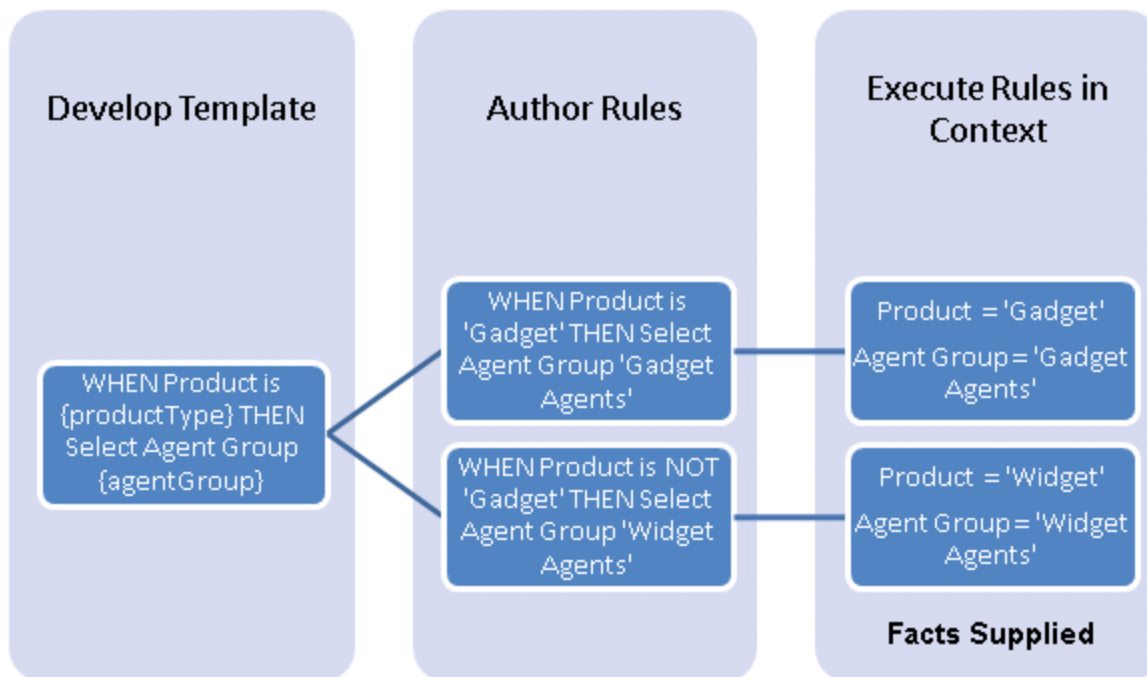
This functionality is enabled by an Eclipse plug-in that can be installed within Composer or in a standalone Eclipse environment.

- To install the plugin, refer to the *Genesys Rule System 8.1 Deployment Guide*. See Chapter 2, Installation.

The plug-in enables developers to create Rule Templates. Rule Templates consist of rule parameters, conditions, actions, and functions. When a Rule Template is published to the Rules System repository, it is made available to be added to Rule Packages. Rule Packages are the deployable objects, which are used to expose rule conditions and actions to business users for creating rules through the Genesys Rules Authoring tool. A brief summary of Rule Templates is presented below. For detailed information, see the *Genesys Rules System 8.1 Rules Development Tool Help*. Once you install the plugin, this help system is available within Composer by selecting **Help > Contents**.

Genesys Rules System Architecture

A logical view of the Genesys Rules System architecture is shown below.



- The first category reflects Rule Template creation, which can be done in Composer if the set of plugins is installed.
- The second category reflects rule creation by business analysts in the Genesys Rules Authoring Tool.
- The third category reflects rule evaluation by the Genesys Rules Engine using the Business Rule block once the **Facts** are known.

Type of Rules

The Genesys Rules System supports both basic and decision table rules.

Basic Rule

A basic or linear business rule is of this form: `WHEN {condition} THEN {action}`. In other words, when the condition is true, the action will occur. This is a rule template. When a business analyst uses the Genesys Rules Authoring Tool to customize a template with valid values, this creates a business rule. The following rules are all valid instances:

- `WHEN Product = 'Gadget' THEN Select Agent Group 'Gadget Agents'`
- `WHEN Product = 'Widget' THEN Select Agent Group 'Widget Agents'`
- `WHEN Customer Segment = 'Gold' THEN Assign Credit Limit '200000'`

This form of rule is preferred for simple actions, such as assigning a value to return back to the application.

Decision Table Rule

A business rule can also take form of a decision table. For example, assume in a particular scenario that there are 3 customer levels: Gold, Silver, and Bronze. For each of these levels, we wish to make an offer to customers based on a qualifying purchase they may have made. Gold customers automatically qualify for a Premium Offer. Silver customers need to have spent \$1000 or more to qualify for that offer, otherwise they get the Special Offer. Finally, Bronze customers need to have spent \$5000 or more for the Premium Offer; or \$2000 or more for the Special Offer; otherwise they are informed of the offers available if they make the qualifying purchase level. **Note:** The Genesys Rules Engine cannot execute Rule Templates.

Business Rule Block

See [Genesys Rules System](#) for information on using Composer to execute a rule block. Once the Rule Package (created from [Rule Templates](#)) that you want to work with are deployed to the Genesys Rules Engine, you can use the Business Rule block on the Server Side palette to create voice and routing applications that use business rules. Use this block to have Composer query the Genesys Rules Authoring Tool (GRAT) for deployed packages. For the Rule Package that you specify, Composer will query the GRAT for the Facts associated with the Rule Package. You can then set values for the Facts, call the Genesys Rules Engine for evaluation, and save the results in a variable. **Note:** This last step (evaluation) happens as part of a VXML or SCXML application that Composer developer creates, not as part of Composer. A [business rule preference](#) specifies the Genesys Rules Engine to work with. **Runtime Parameters** The following parameters (defined in Preferences) are used at runtime, when the VXML and SCXML application queries the GRAT to execute the rule.

- grat_username -- a user login for accessing the GRAT server
- grat_password -- the password for the above login
- grat_server -- a URL for the GRAT server, for example: <http://hostname:8080/genesys-rules-authoring>
- grat_tenant -- the tenant associated with the login, e.g. Environment

The Business Rule block has the following properties:

Name Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Block Notes Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Business Rule Package Property

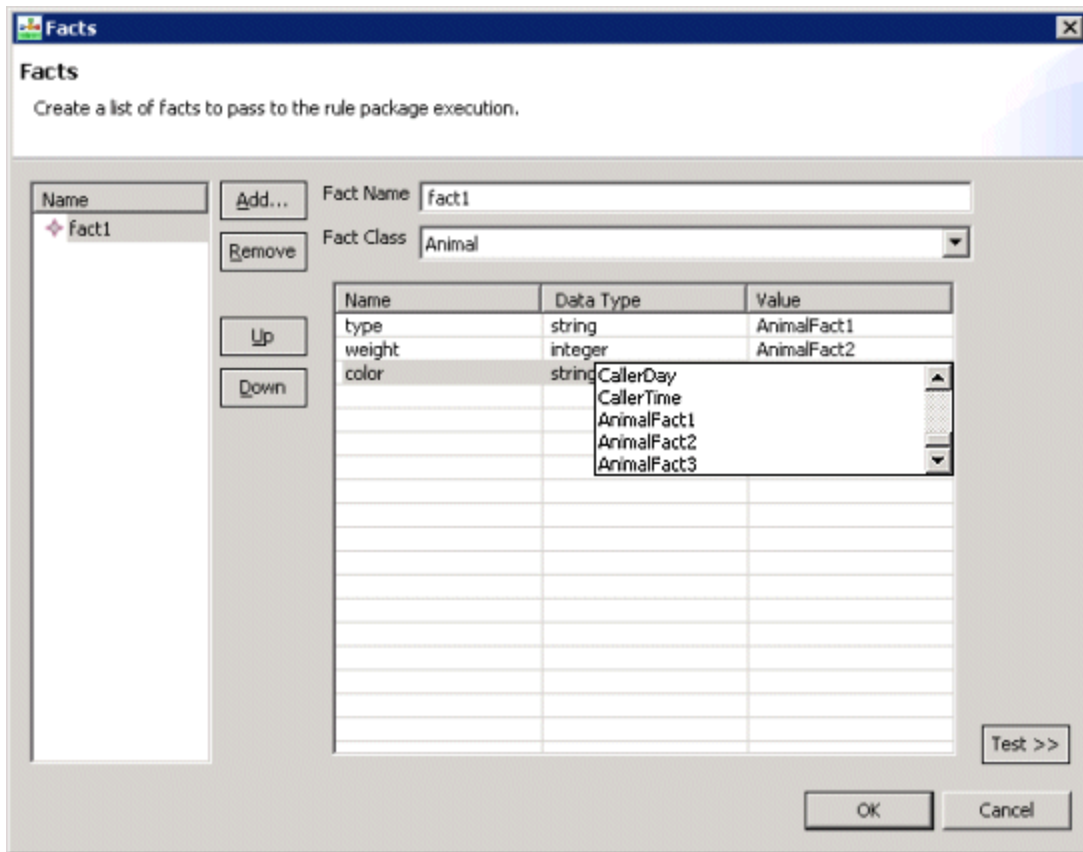
Use to select the Rule Package (collection of related rules) you would like to execute. Packaging rules together allows the business analyst to define which rules will support a particular application. Before using this property, you must set Business Rules Preferences.

1. Click the  button to request Composer to connect to the Genesys Rules Authoring Tool Server using the information specified in **Business Rule Preferences**. After a successful connection, the Business Rule Package dialog appears.
2. Select a Rule Package and click **OK**. The dialog closes and the name of the Rule Package appears under Value.

Facts Property

Use this property to execute the logic contained in the selected Rule Package by supplying input parameters called *Facts*. To specify Facts:

1. Click the **Facts** row in the block's property table.
2. Click the  button to open the Facts dialog box.
3. Click **Add**. The dialog box adds additional fields consisting of the Facts to use when executing the Rule Package. You then click the down arrow and select a value or a variable that contains the value for each Fact. An example dialog box is shown below.



Facts.gif

4. Enter the **Fact Name** field.
5. Click the down arrow and select an entry for the **Fact Class** field.
6. Click the down arrow and select a value or a variable that contains the Fact value.
7. Click **Add** again to enter another Fact, or click **OK** to finish.

Delete Button To delete a Fact:

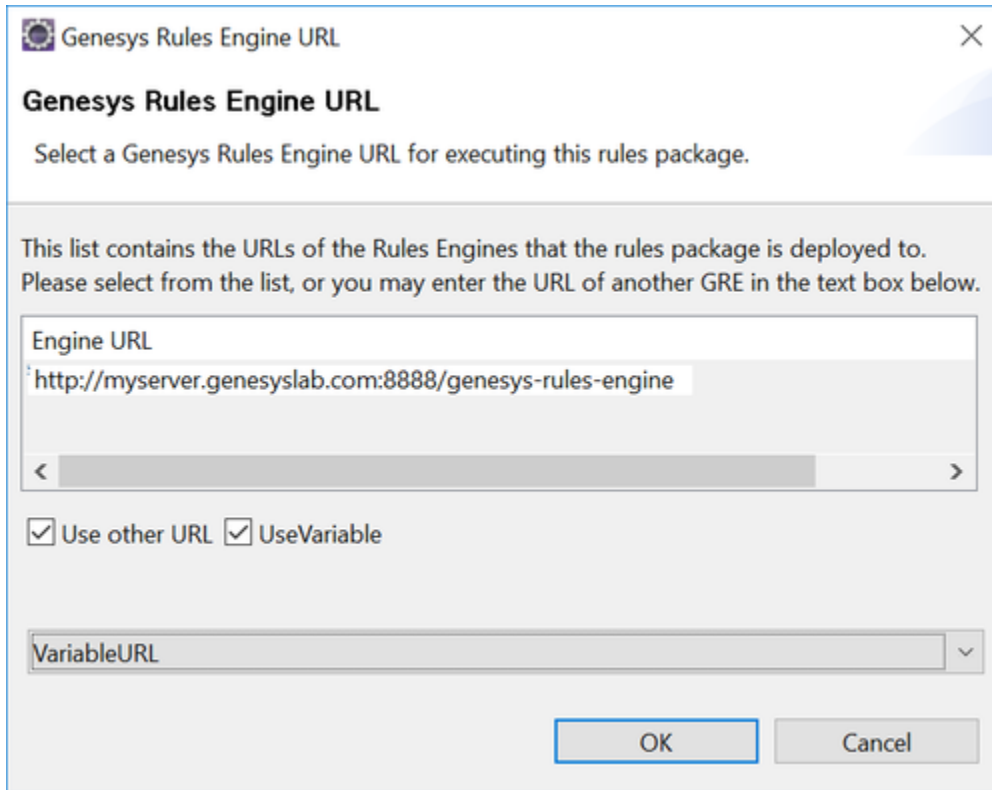
1. Select an entry from the list.
2. Click **Delete**.

Rules Engine URL

Select the variable containing the Genesys Rules Engine URL. Background: Starting with 8.1.2, Composer-generated applications no longer interact with the GRAT server at runtime. Previous requests to the GRAT Server were done to retrieve the URL of the GRE server to which a rules package is deployed. Instead, the runtime applications now use the Rules Engine URL property, which is passed into the application via the IVR Profile or an Enhanced Routing Script object. You can use this Rules Engine URL property to override any GRE URL configured in the IVR Profile or

EnhancedRouting Script object.

Starting with Composer 8.1.440.18, you can specify a variable when the **Use Variable** option is selected.



The screenshot shows a dialog box titled "Genesys Rules Engine URL". Inside, there is a section titled "Genesys Rules Engine URL" with the instruction "Select a Genesys Rules Engine URL for executing this rules package." Below this, a text box contains the URL "http://myserver.genesyslab.com:8888/genesys-rules-engine". Underneath the text box are two checked checkboxes: "Use other URL" and "UseVariable". At the bottom, there is a dropdown menu labeled "VariableURL" and two buttons: "OK" and "Cancel".

This can be used for testing in complex deployments where, for example, the GRE load balancer (or respective instances) is not accessible from the network zone where Composer is located. This option provides the possibility to configure the URL for testing the Rules.

Exceptions Property

The Business Rule block supports the following exceptions. They correspond to the HTTP status codes returned by the Business Rule Server (BRS).

Exception Event Name	HTTP Return Code	BRS Error Code	Description
error.com.genesyslab.com-400	400 - badrequest	610	The received URI does not match the Engines REST specification.
error.com.genesyslab.com-404	404 - notfound	620	The package for the evaluation request received was not found.

error.badfetch.http			Any other HTTP error.
error.com.genesyslab.composer.error.notacceptable	406	602	The evaluation request received could not be converted to a valid knowledgebase-request message, or the evaluation request received could not be evaluated due to an exception.

Details of the exception can be obtained from the body of the response. The Composer application will log the description. The JSON body of the response will look like the following:

```
{  error:{      code:6xx,      description:error message  } }
```

Also see [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Condition Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Logging Details Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Log Level Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Enable Status Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Interaction ID Property

Set to a meaningful value or keep the default value, which is the system variable InteractionId. Applicable to workflows only. Can be used for "interaction-less" processing for scenarios where the

InteractionId variable is not automatically initialized, but instead must wait for an event. An example would be an SCXML application triggered by a Web Service that does not add an interaction. Background: Previous to 8.1.1, Composer did not expose an Interaction ID property. Instead, when ORS started processing an interaction, a generated SCXML application automatically initialized the system variable, InteractionId. This variable was then used internally by Routing and certain eServices blocks when interacting with ORS. With the introduction of support for Interaction-less processing, you can now define a specific event (**IPD Events property**) to initialize InteractionId, or not define an event at all. For scenarios with an interaction (IPD Diagram/Wait For Event=interaction.present for example), you may keep the default value for the Interaction ID property. The default value is the system variable InteractionId, which is initialized automatically in this case. For other scenarios (any scenario where the system variable InteractionId is not set), you may choose to:

1. Not use blocks that require an Interaction ID
2. And/or set the Interaction ID property to a meaningful value
3. And/or assign a meaningful value to the InteractionId system variable

Output Result Property

Use this property to save the results of the business rule execution to a variable. To select a variable:

1. Select the **Output Result** row in the block's property table.
2. In the Value field, select one of the available variables from the drop-down list. Does not need to match the variable name that is coming back as a result of the web request.

The format of returned data is JSON. Any post-processing work to be done on returned results can be done in the existing **Assign block** which provides access to ECMAScript functions. It supports writing simple or complex expressions to extract values out of JSON strings and arrays. In a workflow, the Output Result can be attached to User Data. In the Specify Output Result Location dialog box, select User Data or Variable. If User Data is selected, the specified name is used as a prefix of the keys that will be added to user data. For example, if you specify abc, then the User Data will look like:

```
'abc_fact1'(list) '@class'      'com.genesyslab.animals.Animal'
                  'color'       'red'
                  'type'        1903
                  'weight'      123
'@class'          com.genesyslab.animals.Car'
'mazda'
```

```
'abc_fact2'(list)
'make'
```

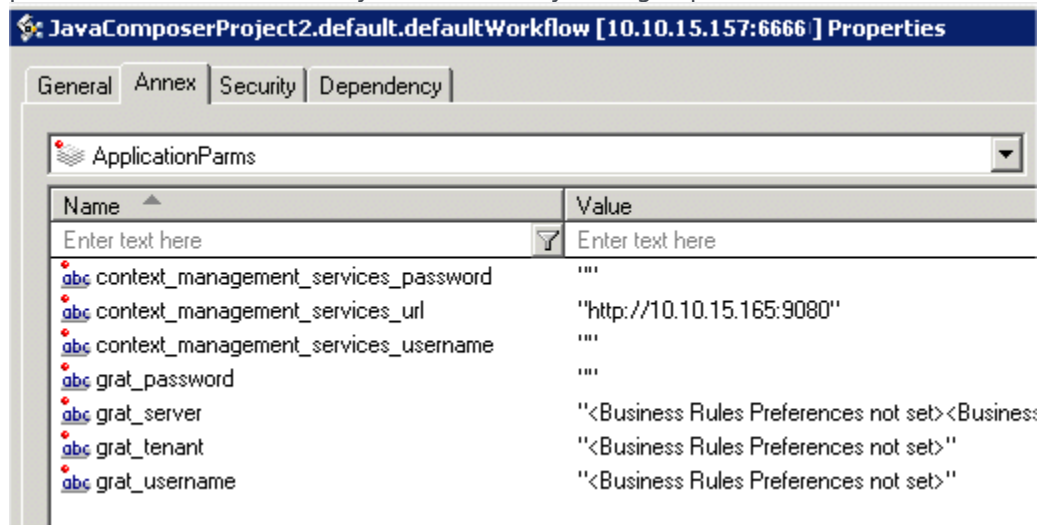
Note: The Output Result property takes effect only during application runtime. Its purpose is to take the output of the rule execution (at runtime) and store returned results back in the specified application variable so other parts of the application can access the data.

Business Rules Block Runtime Configuration

The table below shows the parameters that must be set up in Genesys Administrator in order for the Business Rules block to work.

	ERS Object Key Names	IVRprofile Object Key Names
GRS	grat_server	grat_server
	grat_tenant	grat_tenant
	grat_username	grat_username
	grat_password	grat_password

The figure below shows an example Enhanced Routing Script object created by Composer. It creates these parameters in the ApplicationParms section in the Annex, so you do not have to key in parameter names. Note: If you accidentally changes parameter names, these functions will not work.



Working With Returned Data

Below is an example on how to work with data returned by the Business Rules block. A sample of the output can look like the snippet below, which will be stored in the output variable myOutputVar.

```
myOutputVar = '{
  'knowledgebase-response':{
    inOutFacts:{
      named-fact:[{
        fact:{
          '@class':"abc.sample2._GRS_Environment",
          businessContext__Level1:"Raleigh",
          phase:"prioritization"
        },
        id:"environment"
      },
      {
        fact:{
          '@class':"abc.sample2.Caller",
          disposition:true
        },
        id:"ourCaller"
      }
    ]
  }
}
```

To extract the value of the disposition field, an expression like this can be used:
`myDisposition = myOutputVar["knowledgebase-response"].inOutFacts["named-fact"][1].fact.disposition`
This will return true.

ORS Extensions Property

Starting with 8.1.4, Composer blocks used to build routing applications (with the exception of the Disconnect and EndParallel blocks) add a new **ORS Extensions** property.